

2. คิว (Queue)

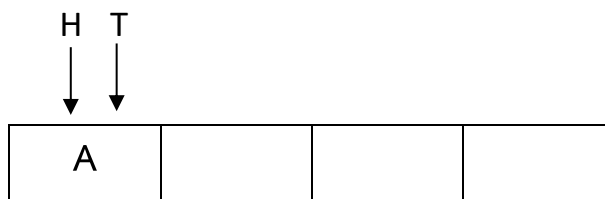
คิวเป็นโครงสร้างข้อมูลที่จัดการข้อมูลตามลำดับ ข้อมูลที่เข้ามาในคิวก่อนจะถูกดึงออกมาก่อน เหมือนการเข้าแถวรับบริการที่ผู้เข้าแถวก่อนจะได้รับการบริการก่อน หากเข้ามาทีหลังก็จะได้รับการบริการทีหลัง เรียกลักษณะการทำงานเช่นนี้ว่า first in first out (FIFO)

การสร้างคิวใช้พอยต์เตอร์ชี้ 2 ตำแหน่ง คือ พอยต์เตอร์ตัวแรกชี้ที่ตำแหน่งที่จะเพิ่มข้อมูล เรียกว่า “tail” ส่วนพอยต์เตอร์ตัวที่สองชี้ที่โหนดแรกที่เป็นตำแหน่งที่จะดึงข้อมูล เรียกว่า “head” ปฏิบัติการกับคิวมีดังนี้

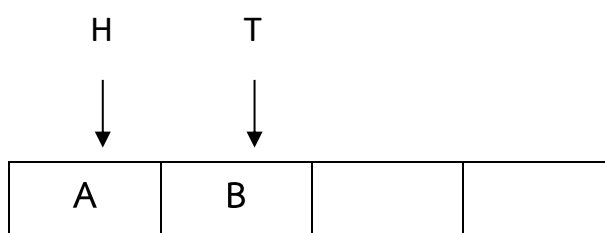
1. การเพิ่มข้อมูลในคิว เรียกว่า **push**
2. การดึงข้อมูลออกจากคิว เรียกว่า **pop**



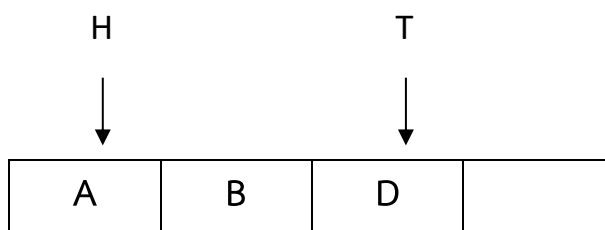
รูปที่ 2.1 เริ่มต้นคิว $H = T = 0$



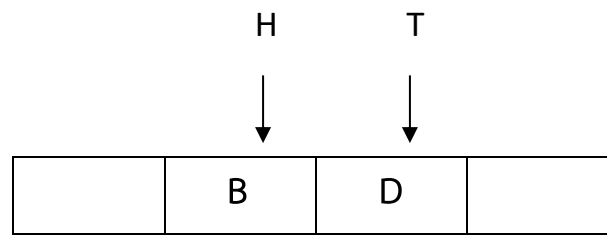
รูปที่ 2.2 หลังการ push A



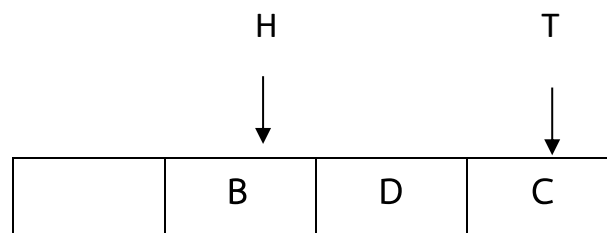
รูปที่ 2.3 หลังการ push B



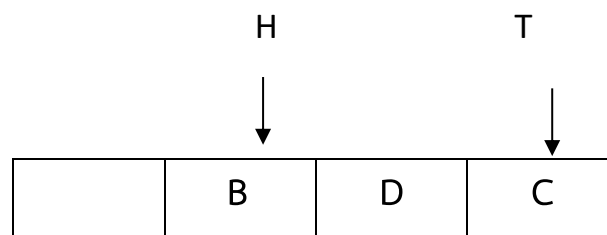
รูปที่ 2.4 หลังการ push D



รูปที่ 2.5 หลังการ pop



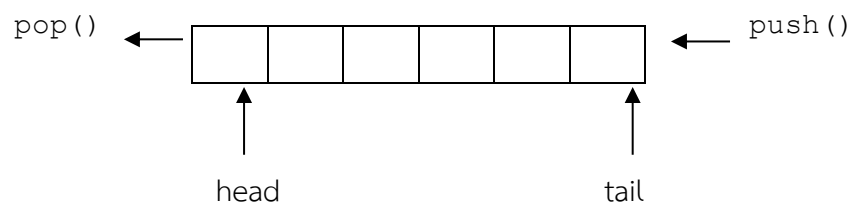
รูปที่ 2.6 หลังการ push C



รูปที่ 2.7 หลังการ push F จะเกิด overflow

โครงสร้างข้อมูล queue ใน STL

ในโครงสร้างข้อมูลแบบ queue ใน STL มีตัวชี้ที่ตำแหน่งแรกสำหรับการดึงข้อมูลออกและมีตัวชี้ตำแหน่งเพื่อเพิ่มข้อมูลเข้าไปในคิว





รูปที่ 2.8 ลักษณะโครงสร้างข้อมูลแบบ queue

การประกาศการใช้ queue และฟังก์ชันที่ใช้

Operation	Description
<code>queue <type> c</code>	Creates an empty list without any elements
<code>c.size()</code>	Returns the actual number of elements
<code>c.front()</code>	Returns the first element (without checking whether a first element exists)
<code>c.back()</code>	Returns the last element (without checking whether a last element exists)
<code>c.push()</code>	Inserts an element into the queue
<code>c.pop()</code>	Removes an element from the queue
<code>c.empty()</code>	Determines whether a queue is empty

ตัวอย่างที่ 2.1 การใช้ operation พื้นฐานของ queue

```
1. #include <iostream>
2. #include <queue>
3. using namespace std;
4. int main()
5. {
6.     queue<string> q;
7.
8.     // insert three elements into the queue
9.     q.push("These ");
10.    q.push("are ");
11.    q.push("more than ");
12.
13.    // read and print two elements from the queue
14.    cout << q.front();
15.    q.pop();  คำ
16.    cout << q.front();
17.    q.pop();  ลบออกจาก Queue
18.
19.    // insert two new elements
20.    q.push("four ");
21.    q.push("words!");
22.
23.    // skip one element
24.    q.pop();
25.
26.    // read and print two elements
```

```
27.     cout << q.front();
28.     q.pop();
28.     cout << q.front() << endl;
30.     q.pop();
31.
32.     //print number of elements in the queue
33.     cout << "number of elements in the queue: " << q.size()
34.     << endl;
35.     return 0;
36. }
37.
```

ผลลัพธ์

```
These are four words!
number of elements in the queue: 0
```

ตัวอย่างที่ 2.2 การวนทำซ้ำเพื่อแสดงค่าใน queue

```
1.  #include <iostream>
2.  #include <queue>
3.  using namespace std;
4.  int main()
5.  {
6.      queue<string> q;
7.      string word;
8.      int n;
9.
10.     cin >> n;
11.     // get elements and store them into the queue
12.     for (int i=0;i<n;i++) {
13.         cin >> word;
14.         q.push(word);
15.     }
16.     cout << "***Output***\n";
17.     // print elements and pop elements out of the queue
18.     while (!q.empty()) {
19.         cout << q.front() << endl;
20.         q.pop();
21.     }
22.     return 0;
23. }
```

ข้อมูลนำเข้า

4
Word1
Word2
Word3
Word4

ผลลัพธ์

Output
Word1
Word2
Word3
Word4

4. Priority Queue

Priority queue เป็นคิวที่จัดลำดับการดึงข้อมูลตามความสำคัญ สำหรับงานบางงานการกำหนดลำดับความสำคัญจะทำให้จัดการงานได้ดีขึ้น

การดำเนินการกับ Priority queue มีดังนี้

1. เพิ่มข้อมูลในคิว insert
2. ดึงข้อมูลออกจากคิว delete (max หรือ min)
3. เข้าถึงข้อมูลในคิว หรือเรียกว่า peek (find max/min)

การประกาศการใช้งาน

1. ใช้กับชนิดข้อมูลพื้นฐานและเป็น priority queue ของค่าที่มากที่สุด

```
priority_queue<type> variable_name;
```

2. ใช้กับชนิดข้อมูลพื้นฐานและเป็น priority queue ของค่าที่น้อยที่สุด

```
priority_queue<type, vector<type>, greater<type>> variable_name;
```

3. ใช้กับชนิดข้อมูลพื้นฐานและเป็น priority queue ของค่าที่ขึ้นอยู่กับฟังก์ชันที่กำหนด

```
priority_queue<type, vector<type>, compare_function> variable_name;
```

ฟังก์ชันที่ใช้

Operation	Description
c.push()	Inserts a new element
c.pop()	Removes the first element
c.top()	Accesses the largest element
c.size()	Returns the actual number of elements
c.empty()	Determines whether a queue is empty

ตัวอย่างที่ 4.1 การเพิ่มสมาชิกใน priority queue ด้วย push()

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      priority_queue<int>pq;
6      pq.push(6);
7      pq.push(1);
8      pq.push(3);
9      pq.push(9);
10     pq.push(2);
11     // printing queue
12     while (!pq.empty()) {
13         cout << pq.top() << endl;
14         pq.pop();
15     }
16     return 0;
17 }
```

ผลลัพธ์

9
6
3
2
1

ตัวอย่างที่ 4.2 การเข้าถึงสมาชิกใน priority queue ที่มีค่ามากที่สุดด้วย top()

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      priority_queue<int>pq;
6
7      pq.push(6);
8      pq.push(1);
9      pq.push(3);
10     pq.push(9);
11     pq.push(2);
12
13     // access the largest element
14     cout << pq.top();
15     return 0;
16 }
```

ผลลัพธ์

9

ตัวอย่างที่ 4.3 การดึงสมาชิกออกจาก priority queue ที่มีค่ามากที่สุดด้วย pop()

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4
5  int main() {
6      priority_queue<int>pq;
7
8      pq.push(6);
9      pq.push(1);
10     pq.push(3);
11     pq.push(9);
12     pq.push(2);
13
14     pq.pop();
15     pq.pop();
16
17     // printing queue
18     while (!pq.empty()) {
19         cout << pq.top() << endl;
20         pq.pop();
21     }
22
23     return 0;
24 }
```

ผลลัพธ์

3
2
1

ตัวอย่างที่ 4.4 การใช้ priority queue กรณีที่ดึงค่าน้อยที่สุด

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      priority_queue<int,vector<int>,greater<int>>pq;
6      pq.push(6);
7      pq.push(1);
8      pq.push(3);
9      pq.push(9);
10     pq.push(2);
11     // printing queue
12     while (!pq.empty()) {
13         cout << pq.top() << endl;
14         pq.pop();
15     }
16     return 0;
17 }
```

ผลลัพธ์

1
2
3
6
9

ตัวอย่างที่ 4.5 การใช้ฟังก์ชัน size()

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      priority_queue<int>pq;
6      pq.push(6);
7      pq.push(1);
8      pq.push(3);
9      pq.push(9);
10     pq.push(2);
11     cout << pq.size() << endl;
12     return 0;
13 }
```

ผลลัพธ์

5

ตัวอย่างที่ 4.6 การใช้ฟังก์ชัน swap()

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4  int main() {
5      priority_queue<int>pq1;
6      priority_queue<int>pq2;
7      pq1.push(6);
8      pq1.push(1);
9      pq1.push(7);
10     pq2.push(3);
11     pq2.push(2);
12     pq2.push(8);
13     pq2.push(9);
14
15     pq1.swap(pq2);
16
17     cout << "pq1 = " ;
18     while (!pq1.empty()) {
19         cout << pq1.top() << " ";
20         pq1.pop();
21     }
22     cout << "\npq2 = ";
23     while (!pq2.empty()) {
24         cout << pq2.top() << " ";
25         pq2.pop();
26     }
27     return 0;
28 }
```

ผลลัพธ์

```
pq1 = 9 8 3 2
pq2 = 7 6 1
```

ตัวอย่างที่ 4.7 การใช้ priority queue ร่วมกับ struct

```
1  #include <iostream>
2  #include <queue>
3  using namespace std;
4
5  struct Person {
6      int age;
7      float height;
8  };
9
10 struct CompareHeight {
11     bool operator()(const Person &p1, const Person &p2)
12     {
13         return p1.height < p2.height;
14     }
15 };
16 int main()
17 {
18     priority_queue<Person, vector<Person>, CompareHeight> pq;
19     Person data;
20     data = Person{38,150};
21     pq.push(data);
22     data = Person{34,170};
23     pq.push(data);
24     data = Person{44,180};
25     pq.push(data);
26     data = Person{35,140};
27     pq.push(data);
28     while (!pq.empty()) {
29         Person p = pq.top();
30         pq.pop();
31         cout << p.age << " " << p.height << "\n";
32     }
33     return 0;
34 }
```

ผลลัพธ์

```
44 180
34 170
38 150
35 140
```

แบบฝึกหัด

1. ให้แก้ไขตัวอย่าง 4.7 โดยใช้ priority queue เก็บข้อมูลแบบ struct และแสดงข้อมูลโดยเรียงตามอายุจากมากไปน้อย
2. ให้แก้ไขตัวอย่าง 4.7 โดยใช้ priority queue เก็บข้อมูลแบบ struct และแสดงข้อมูลโดยเรียงตามอายุจากน้อยไปมาก